**Jelgenerátorok(háromszög,négyszög)**

Készitette: Szalma Szilárd

Vezető tanár: Dr. Brassai Sándor Tihamér

**1.Terv céljának megfogalmazása:**

Az újrakonfigurálható digitális áramkörök tantárgy keretein belül szeretnék megvalósitani egy jelgenerátort, az általa generált jelet pedig egy interfészen keresztül szeretném kiküldeni és monitorizálni. A jelgenerátor feladata különböző hullámformájú, amplitudójú és frekvenciájú jelek előállitása. Ezek a hullámformák lehetnek: háromszögjel, fűrészfogjel, négyszögjel, szinuszjel vagy akár általános formájú jel is.

**2.Rendszerrel szembeni követelméynek:**

**2.1 Általános követelmények:**

-Legalább két tipusú jel generálása

-A jelek SPI interfészen való kiküldése

-Felhasználóbarát: könnyen kezelhető

-Paraméterek változtathatósága: amplitudó, frekvencia, fázis

**2.2 Specifikus követelmények:**

-Architektúra: Xilinx SoC

-Hardver: FPGA áramkör: Zynq7000

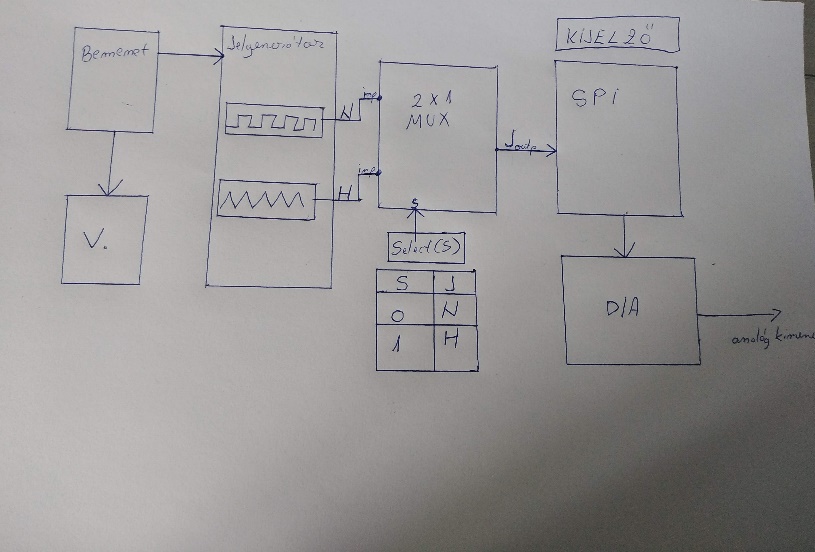
-Tervezési Stratégia: Az FSMD alapú tervezés használata

-Hardver leiró nyelv: VHDL(Very High-Speed Integrated Circuit Hardware Description Language)

-Tervezésre és tesztelésre alkalmazható szoftver eszköz: Vivado 2017.4

**3. A terv működésének rövid leirása:**

Figure 1

****

A figure 1 ábrán látható a rendszerem megvalósitandó tömbvázlata. A képen szerepelnek a bemenetek, a jelgenerátor, az SPI interfész, egy multiplexer, egy D/A átalakitó és egy kijelző.

**Jelgenerátor:** Generálja a szükséges jelformákat: háromszög és négyszög. Kimenete egy 16 bites szám.

**SPI:** Serial Periferial Interface. Soros interfész, ami bitenként továbbitja a generátor által generált értéket a D/A konverter felé.

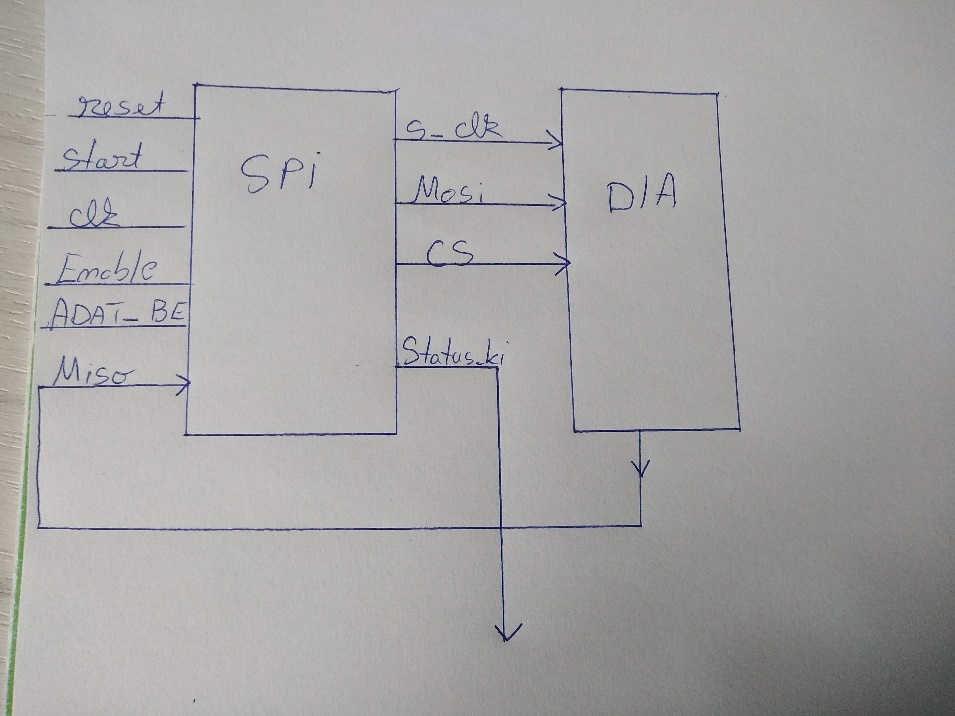
**D/A:** Digitál-analóg konverter. Fogadja a generátortól érkező adatokat és átalakitja analóg jellé.

**Multiplexer:** Kiválasztja a két jel közül az egyiket a Select bit értékének változtatásával.

**Kijelző:** A jel monitorizálása.

**4.SPI ( soros periférikus interfész):**

Figure 2

****

Az SPI vezérlő feladata a generátor által előállitott 16 bites érték, bitenként való kiküldése a digitál-analóg konverter felé, ami a kettes ábrán is látható.

**ADAT\_BE:** Ezen porton kapja meg az interfész a jelgenerátor értékét.

**Mosi(master out slave in):**  Feladata az ADAT\_BE által érkező 16 bit kiküldése a D/A konverterhez. A mosi 1 bites, ezért egyszerre csak 1 bitet képes átküldeni.

**Miso(master in slave out):**  Fogadja a D/A konvertertől visszafele érkező adatokat bitenként.

**S\_CLK:**  Ez az órajel alapján fogadja a D/A konverter az adatokat. Mivel a konverter maximálisan 20 Mhz-el képes dolgozni, az SPI pedig 100 Mhz-el, ezért nem használható a clk, szükség van az S\_CLK generálására.

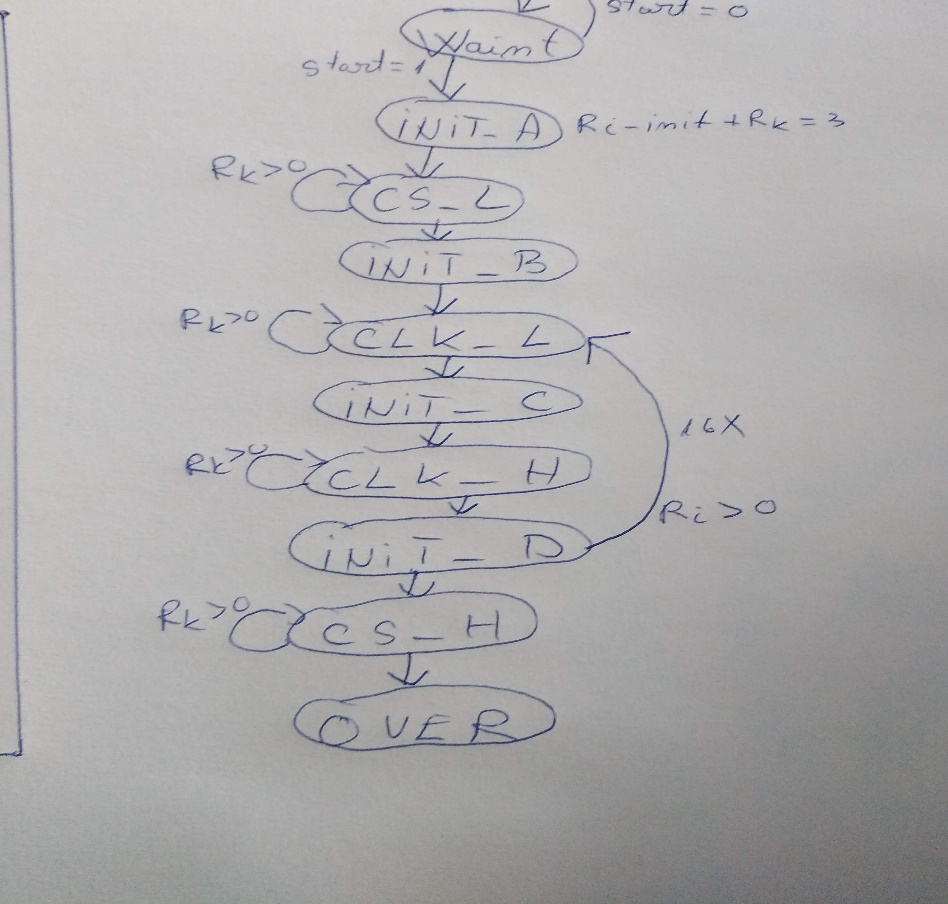
**CS:** Feladata megmondani, hogy mikor legyen aktiv a slave. Értéke kezdetben 1, 0-ra állitással pedig jelzi a slave-nek, hogy hamarosan érkeznek az adatok. Alapértékbe kerülésekor vége van az adatküldésnek.

**Status\_ki:** Jelzi a folyamat végét. Akkor jelez, ha mind a 16 bit átküldésre került.

Az SPI interfész megvalósitásához FSMD alapú tervezést fogok használni. Az FSMD jelentése: Finite State Machine with Datapath. Ebben a tervezési módszerben szükség van megtervezni egy állapot automatát adatutakkal. A komplex állapotgépben megváltoztatjuk bizonyos adatok rétékét. Az SPI-ben szereplő értékeket, és azoknak a változását szét kell választani. Nem lehet egy állapotban a jelenlegi értéket olvasni, és azt meg is változtatni. Ennek érdekében szükség van egy aktuális állapotra,ahol a jelenlegi érték van, és egy következő állapotra, amiben megváltoztatjuk a jelenlegi értéket.

**4.1. Az SPI állapot diagramja:**

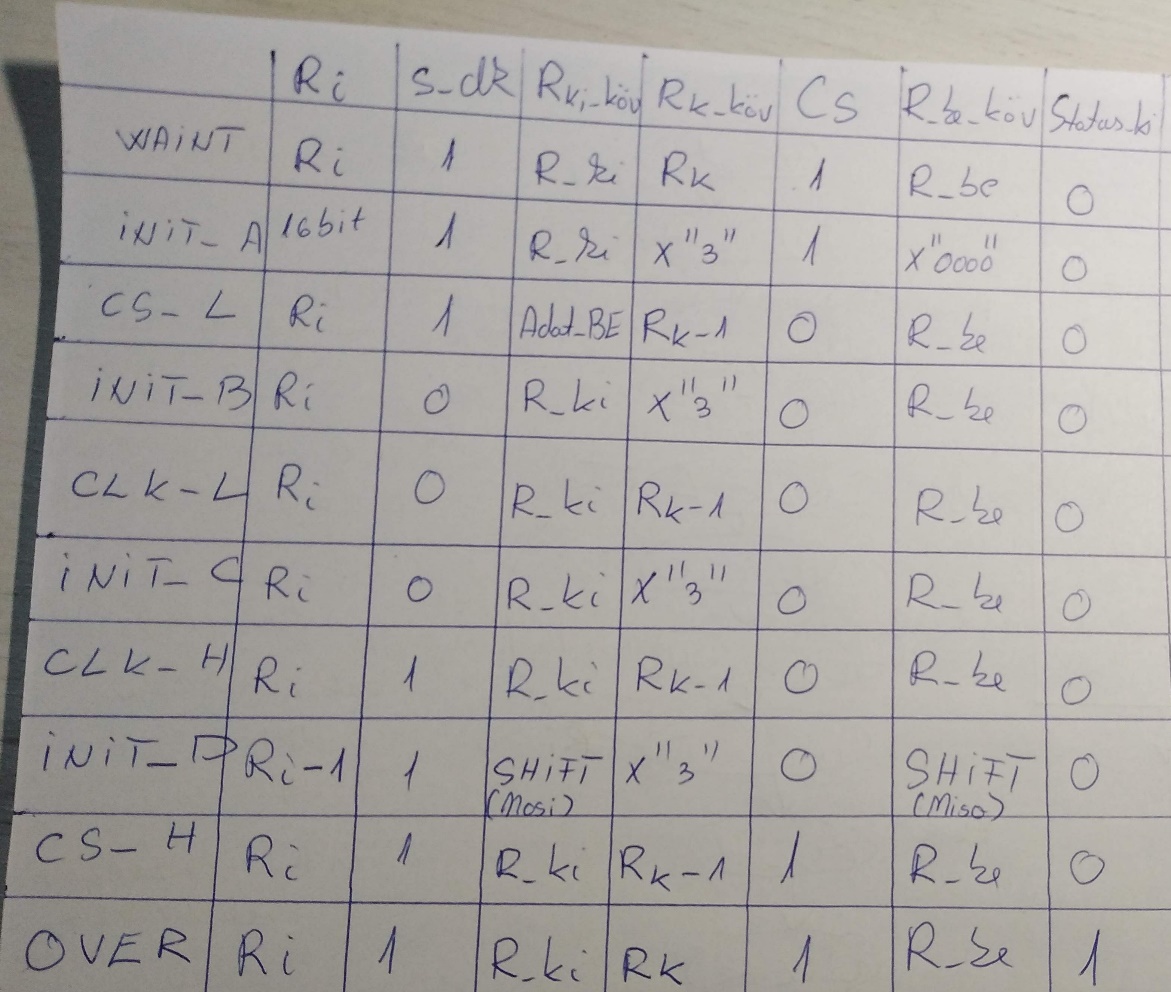
Figure 3

****

A harmadik ábrán láthatjuk az SPI állapot diagramját, ami megmutatja, hogy és milyen feltételek alapján megyünk át egyik állapotból a másikba. Kezdetben a **WAINT** állapotban vagyunk, egészen addig amig a start jelünk átvált nulláról egyre. A WAINT állapotból átmegyünk az **INIT\_A** állapotba a start jel hatására, ahol az Ri inicializálódik 16-ra és az Rk-t beállitjuk 3-ra. Az INIT\_A-ból átmegyünk a **CS\_L**-be, ahol addig maradunk, amig az Rk értéke nagyobb mint nulla. Ha nullázódik, a chip select(CS\_L) nulla lesz, azaz kezdődik az adatküldés. Ugyanez a folyamat megtörténik a következő állapotokban is. A **CLK\_L**(low) állapotban nullázzuk az órajelet, mig a **CLK\_H**(high) állapotban visszarakjuk egyesre. Minden CLK állapotban addig maradunk amig az Rk nagyobb mint nulla, ahogy ezt az ábra is mutatja. A CLK\_H állapotban megtörténik a kiküldés, amit az **INIT\_D** állapot követ, ahol a SHIFT-elés történik. A CLK\_H és az INIT\_D állapotok által közrefogott ciklust 16x ismételjük meg, azaz addig amig az Ri nulla nem lesz, tehát az adat mind a 16 bitje kiküldésre került. Ha ez megtörtént, a **CS\_H** (high) állapotba kerülünk, ahol majd a chip select vissza kerül egyesbe, a folyamat befejeződött. Átkerülünk az **OVER** állapotba, és minden kezdődik elölről.

**4.2 Az SPI állapotjainak táblázata:**

Figure 4



A fenti táblázatban láthatjuk, hogy a regiszterekben levő érték, hogy változik, végighaladva az állapot diagram ciklusán.

**Ri\_köv:** Számláló, amit INIT\_A-ban inicializálok 16-ra. Mivel CLK\_H-kor megtörténik az elküldés, az INIT\_D állapotban kivonunk egyet az Ri értékéből. Más állapotokban az Ri\_köv értéke megegyezik az Ri-vel.

**S\_CLK:** Saját órajel, 20 MHz frekvencián. Mivel saját magam generálom ki az órajelet, én mondom meg hol legyen felfutó és lefutó él. Hogy szimmetrikus legyen, az első 3 állapotban az értéket 1-re raktam, a következő háromnál 0-ra, a maradék négynél pedig visszaraktam 1-re.

**R\_ki\_köv:** Az R\_ki\_köv regiszternek két fontos szerepe van. CS\_L állapotban betöltődik az ADAT\_BE, azaz a 16 bites szám. CLK\_H állapotban megtörténik a 16 bit 0-dik bitjének kiküldése. Ezért mivel már azt többet nem kell kiküldeni, a következő, azaz INIT\_D állapotban el SHIFT-eljük jobbra mind a 16 bitet, hogy a 0-dik elem helyére az első elem kerüljön.

**Rk\_köv:**  A negyedik ábrán látható, hogy az INIT állapotokban, a regiszter megkapja a 3-t, a CLK és CS állapotokban, pedig kivonunk egyet belőle. Ezzel szavatoljuk az órajel leosztást 100-ról 20 MHz-re.

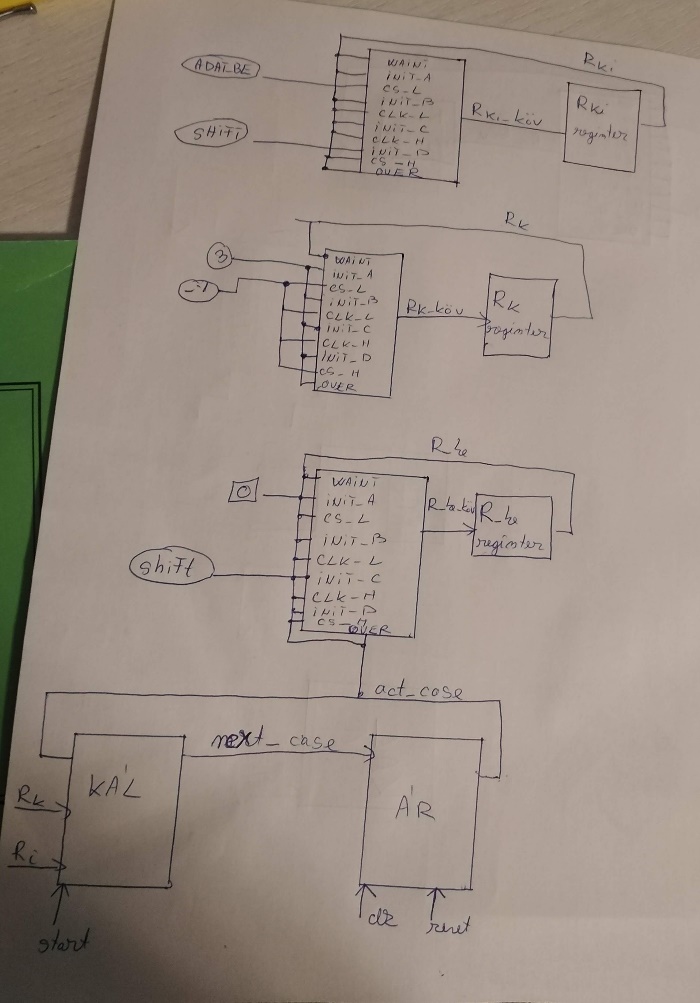
**CS:** A cs regiszter feladata jelezni a D/A konverternek, hogy kezdődik az adatküldés. CS\_L állapotban 1-ről 0-ra vált, ezzel jelezve. Értéke mindaddig 0 marad, mig az adatküldés folyamatban van. Az értéke a CHIP\_HIGH állapotban visszaállitódik 1-re, mint azt a neve is sugallja, ezzel jelezve az adatküldés végét.

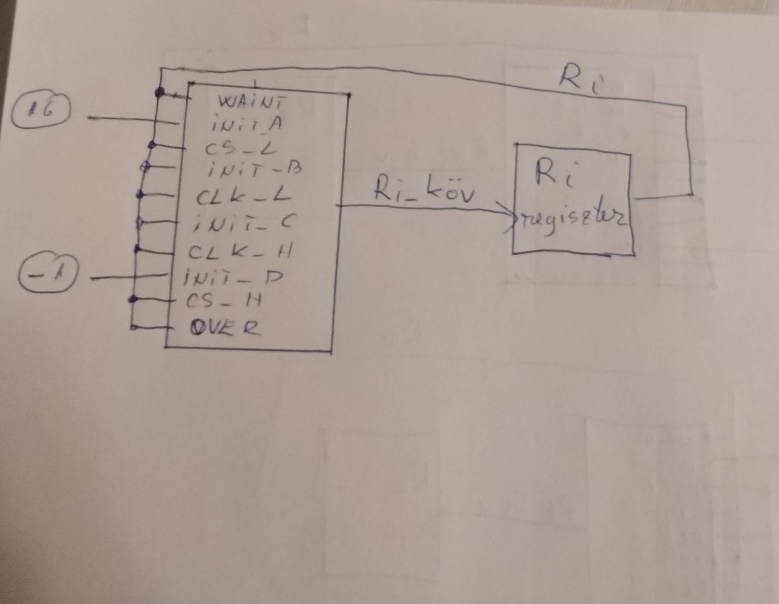
**R\_be\_köv:** Ez a regiszter az én tervem kivitelezésében nem hasznáaltos, mivel nálam a D/A konvertertől nem érkezik visszafele információ. Az INIT\_A állapotban érték adódik neki, majd pedig megtörténik a Miso-ba való SHIFT-elés. A Miso, hasonlatos a Mosihoz, a vissza érkező adatokat továbbitja bitenként ellenkező irányba.

**Status\_ki:** A status\_ki regiszterem szerepe jelezni a folyamat végét. Értéke mindvégig 0, csak az OVER állapotban változik az értéke 1-re, ezzel jelezve, hogy mind a 16 bit elküldésre került. A status\_ki regiszternek nagyon fontos szerepe van, ha új adatot akarok küldeni, hisz ha a status\_ki összekapcsolódik a jelgenerátorral, a generátor tudja, mikor küldheti a következő adatot.

**Összegezve a fenti rajzot és a táblázatot:** az állapot diagram alapján megirható a következő állapot logika VHDL kód, a táblázat alapján pedig a with selectek VHDL kódja.

**4.3. Az SPI áramköri rajza:**

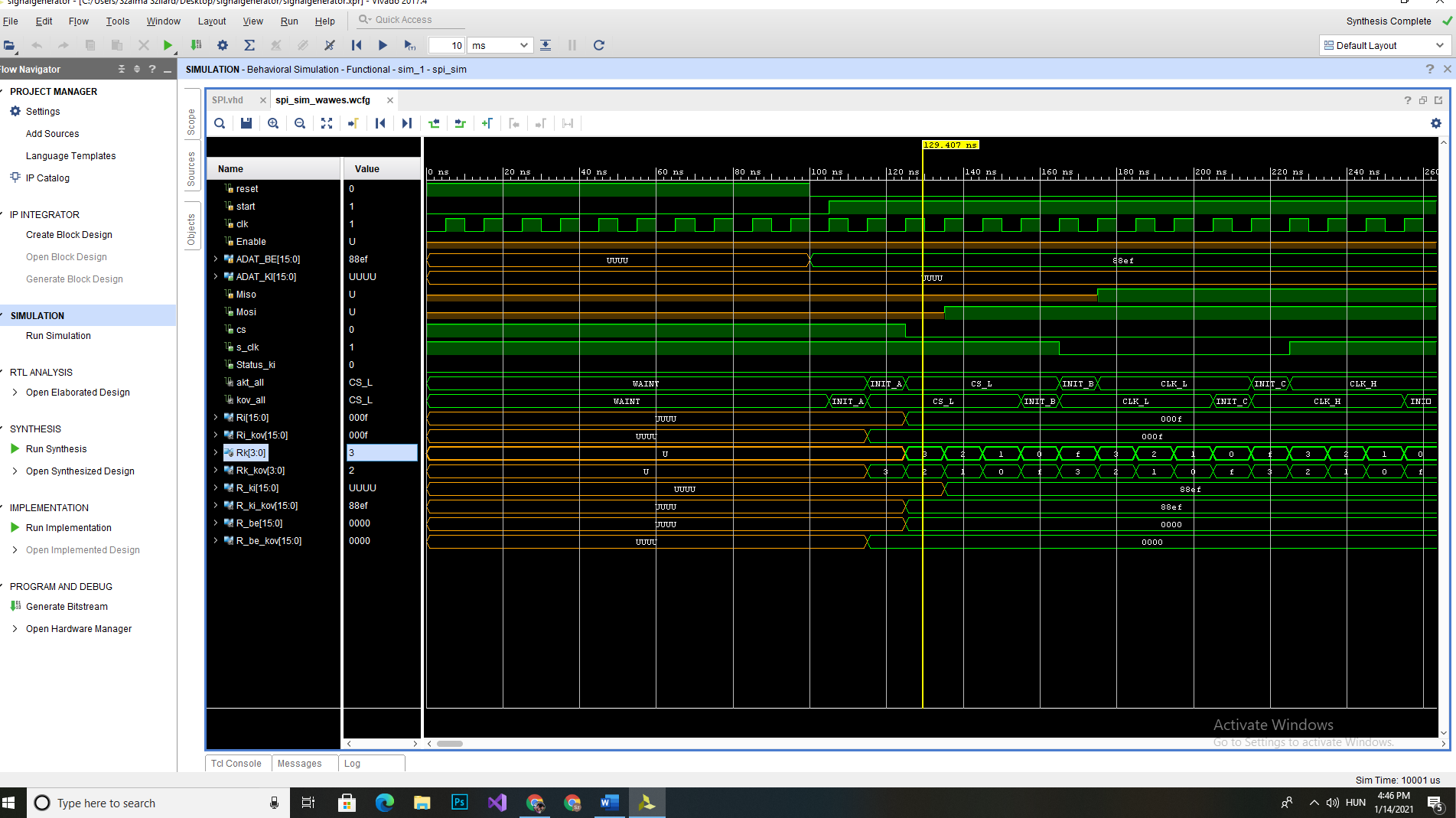
****

****

(Nem sikerült összekötni a regisztereket)

**4.4 Az SPI szimulációja:**

Figure 5

****

A fenti ábrán látható az SPI szimulációja.

**A szimuláción látható:**

-CS\_L állapot 4 órajelig tart, előzőleg az INIT\_A 1 órajelet. Igy kijön az 5 órajel.

-amikor a start egyesbe vált, megkezdődik az INIT\_A állapot.

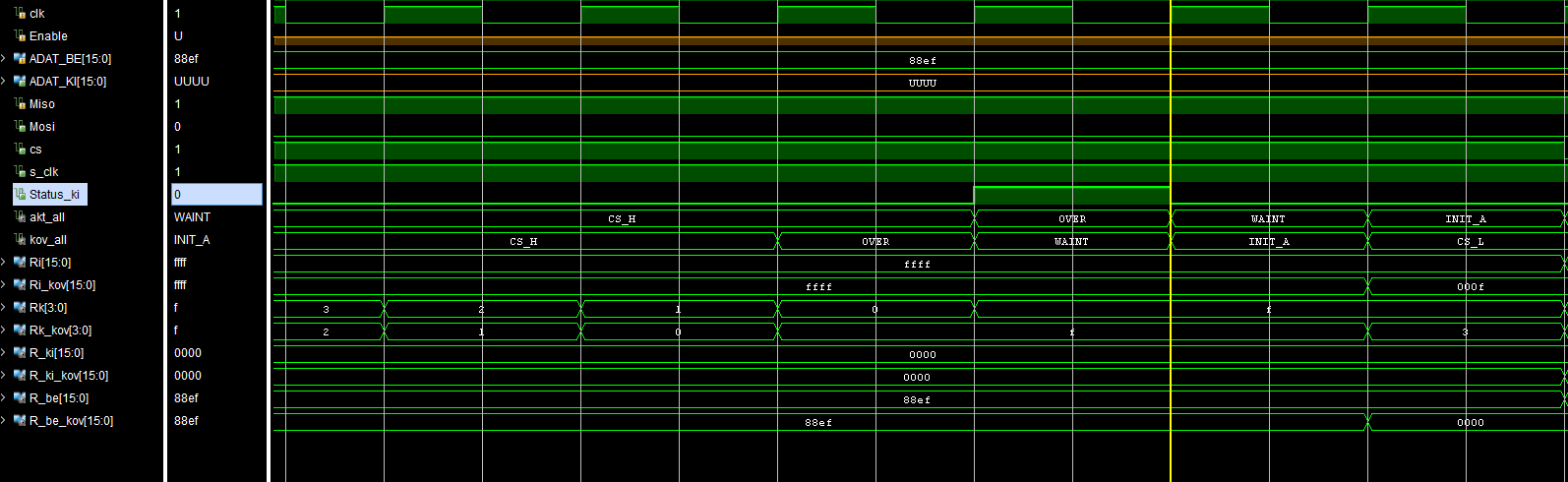
-az ADAT\_BE-be betöltődik a 88ef.

-az állapotok követik egymást: WAINT, INIT\_A, CS\_L, stb.

-Az Rk értéke 3-tól esik visszafele, egész -1-ig, amit a szimuláció f-el jelöl.

- A Cs, chip select, látható, hogy 1-ről 0 lesz, kezdődni fog az adatküldés.

Figure 6



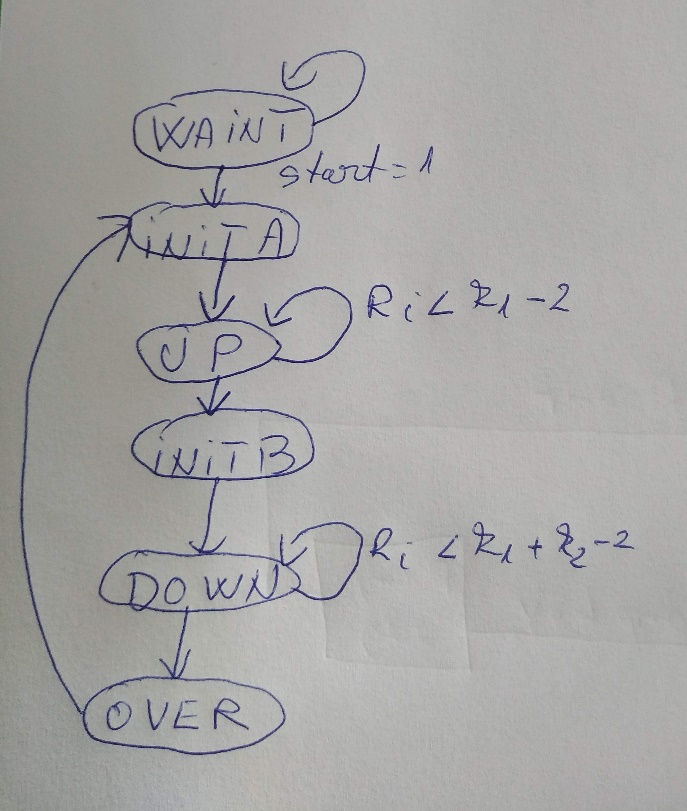
A hatodik ábra a szimuláció megközelitett képét mutatja. Látható, hogy az adatküldés után, a Status\_ki jel, 0-ból 1-re vált, ekkor az aktuális állapotom már az OVER-ben van, a következő állapot meg újra az WAINT lesz. A szimuláció úgy műküdik, hogy a Misonak is szerepe lesz. A szimulációban, amit a Mosi kiküld a D/A konverternek, azt a Miso vissza is küldi. Látható, hogy mikor a Status\_Ki 1-be vált, az R\_be ugyanazt a **88ef** értéket tartalmazza, mint ami az ADAT\_BE értéke volt.

**5. A háromszögjel**

Az egyik általam generált jel a háromszög jel. A háromszög jel két részből tevődik össze. Van egy lemenő és egy felmenő része.

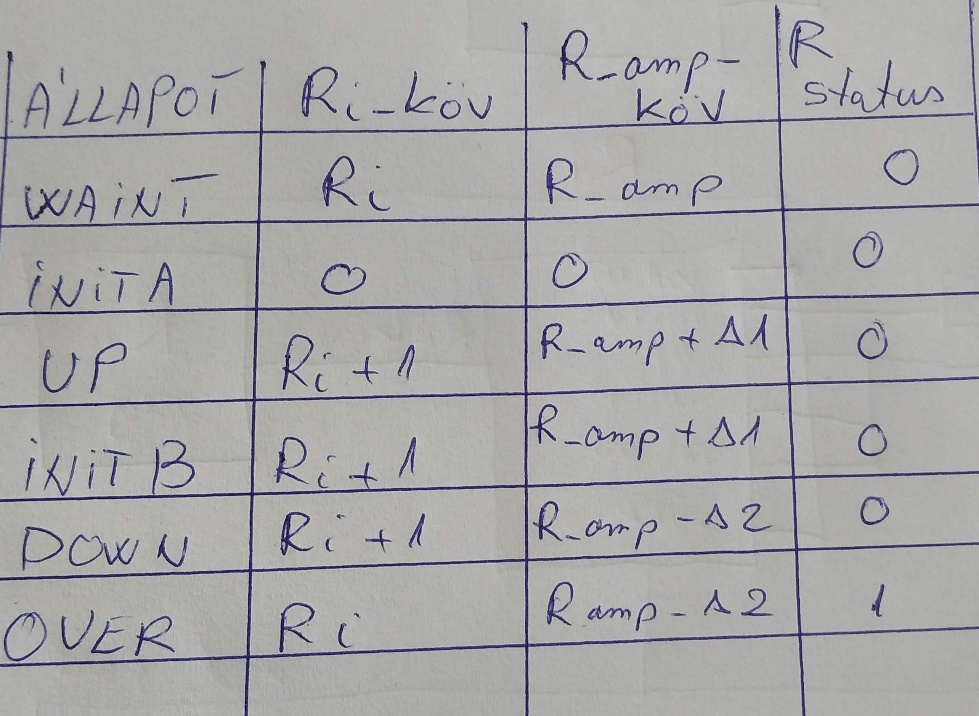
**5.1 A háromszögjel állapot diagramja**

Az alábbi állapot diagramon is megfigyelhető ez a két rész. Mint ahogy megszoktuk, a WAINT állapotból akkor lép ki a rendszer amikor megkapja a start jelet. Két fontos állapot van, az UP és a DOWN, ennek a két állapotnak a segitségével rajzolódik ki a háromszög idézőjeles jobb és bal fele. Látható, hogy ezekben az állapotokban addig marad a rendszer, amig az Ri<k1-2 és Ri<k1+k2-2 feltételek teljesülnek. A “k” jelentése, hogy hány lépésből álljon a felmenő és lemenő rész. Az utolsó over állapottal véget ér az állapot diagram, és keződik elölről, kihagyva a WAINT állapotot.



Figure

**5.3 A háromszögjel állapotjainak táblázata:**

****

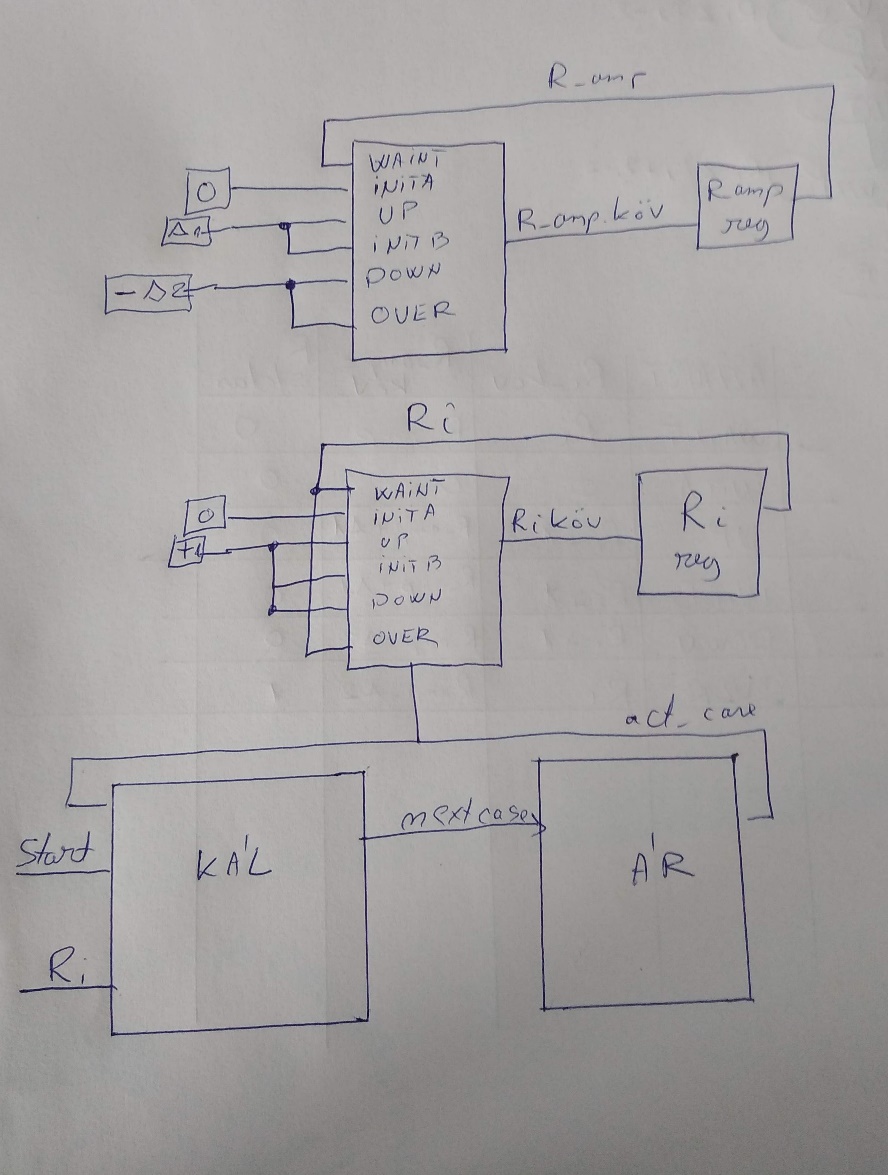
Figure

**Ri\_köv:** Számláló, meghatározza hogy mennyi ideig legyünk az UP és DOWN állapotokban. Az állapot diagramon is látható volt, hogy addig maradunk az UP-ban amig a számláló kissebb mint a k1. Az INIT\_A állapotban megtörténik az inicializálása a számlálónak, a többi állapotban növelünk.

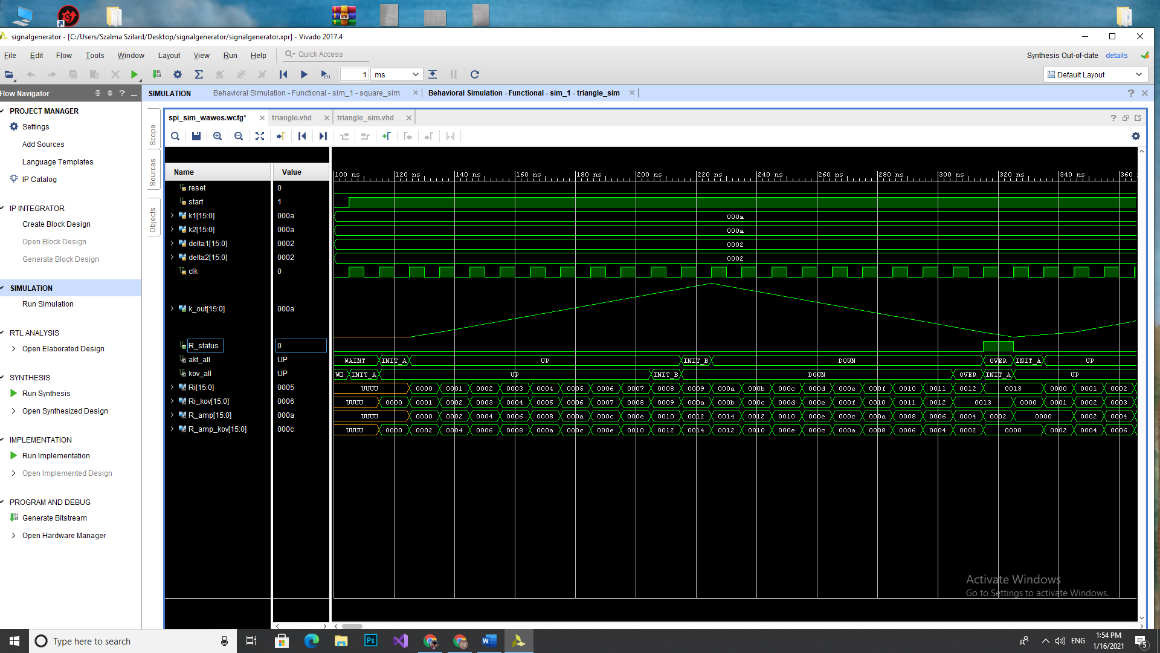
**R\_amp\_köv:** Ebben tárolódik az amplitudó értéke. Itt is az amplitudó értéke az INIT\_A állapotban inicializálódik. Az UP állapotban a jelenlegi amplitudóhoz, hozzzáadódik a delta értéke, a DOWN állapotban pedig kivonódik.

**R\_status:** Értéke mindig 0, amig a folyamat véget nem ér. Ha kirajzoltuk a jelet, értéke 1-re vált.

**5.4 A háromszögjel áramköri razja:**

****

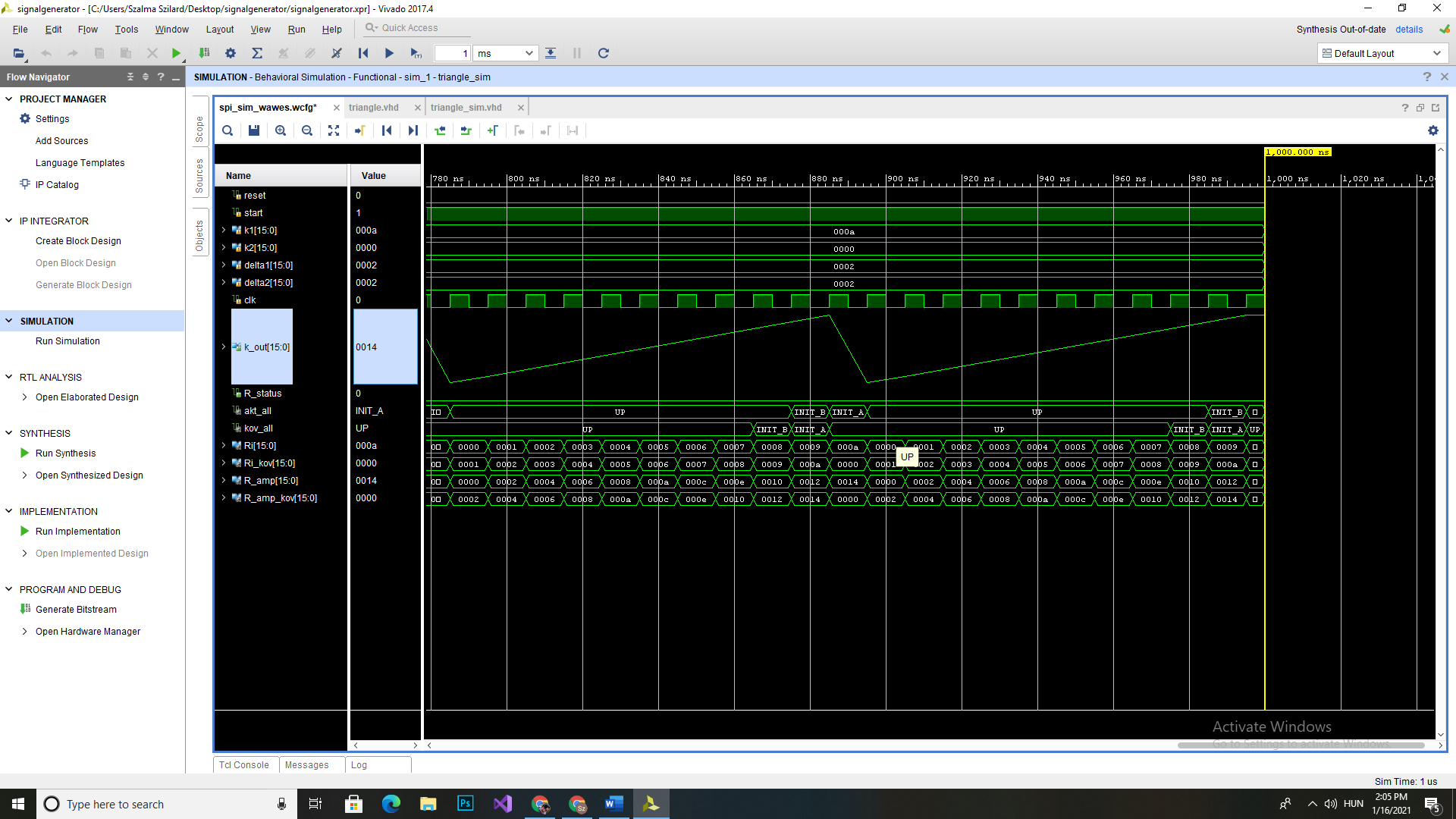
**5.5 A háromszög jel szimulációja:**

****

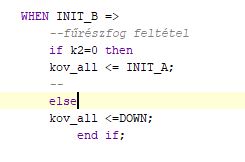
Figure

A kilecedik ábrán látható a háromszögjel szimulációja. Látható, hogy mivel a k-nak 10-et adtam meg, ezért a számláló 10-ig számol és adja hozzá fokozatosan az amplitudót. Amikor a számláló elérte a k-t, vagyis felértünk a csúcsra, a DOWN állapot kezdődik meg, amikor is a delta2-t vonjuk ki az amplitudóból, igy csökken a jelünk lefele. Látható, hogy miután a jel kirajzolódik, a status regiszterem felugrik 1-be, jelezve a folyamat végét.

Mivel a számláló a k értékétől függ, ezért a háromszögjel annál elnyújtottabb, minél nagyobb a k értéke. Ebbe belegondolva, ha a háromszögjel bal felének kirajzolásához szükséges k2-t lenullázom, amig a k1 pl 10-en marad, akkor a megnyúlás 0 lesz, és a jel egyből leesik 0-ba. Ha ez megtörténik, a háromszögjelből fűrészfog jelünk lesz, ami látható az alábbi 10-es ábrán.



Figure

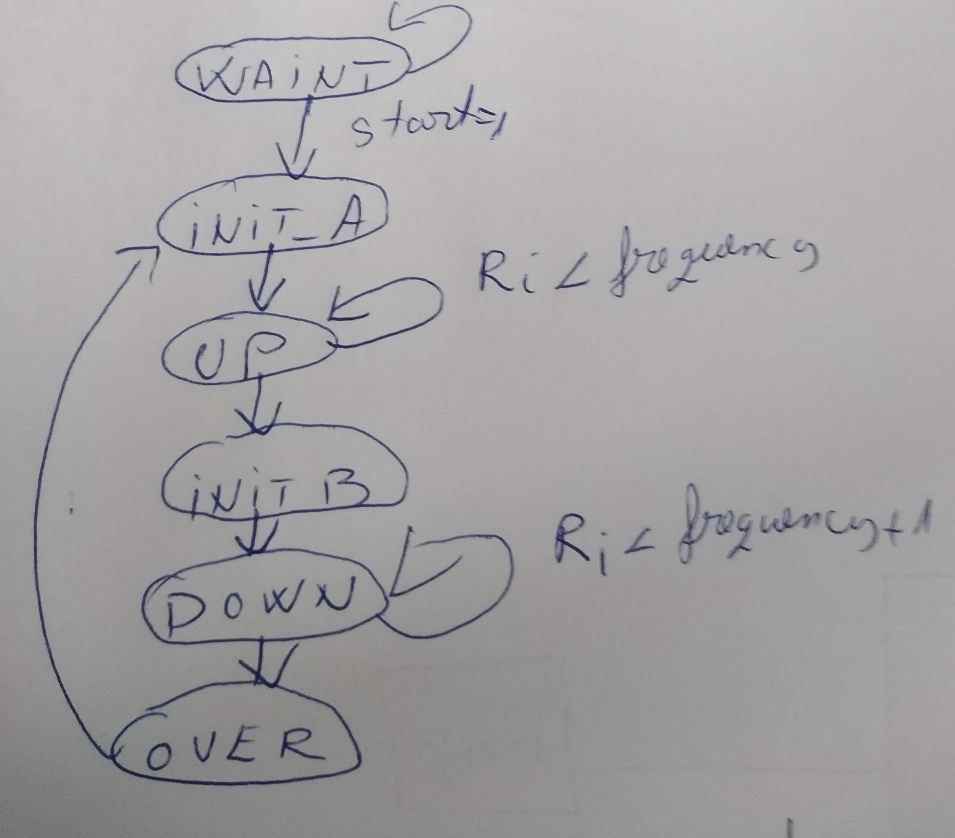


Figure

**6. A négyszögjel**

A négyszögjel hasonlatos a háromszög jelhez, nem kell sokat változtatni, hogy a háromszögből négyszög jelet kapjunk. A négyszögjel létrehozása frekvencia és amplitudó függő. A négyszögjelnél megjelenő állapotok ugyan azok, mint a háromszög jelnél. Ezt láthatjuk az alábbi 12-s ábrán.

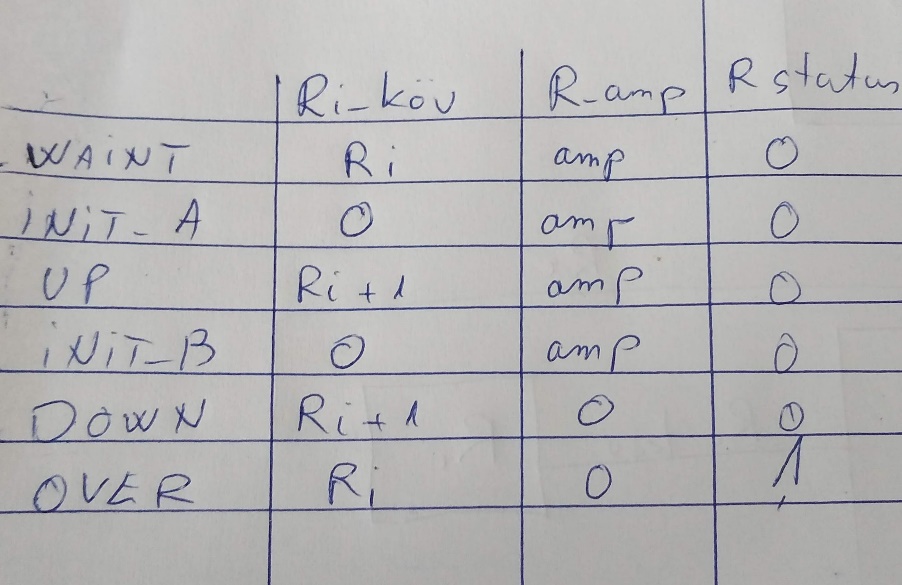
**6.1 A négyszögjel állapot diagramja**

****

Figure

Az állapotok megegyeznek a háromszög jel állapotaival, azzal a kivétellel, hogy a frekvenciától függ, hogy meddig maradunk az UP és DOWN állapotokban.

**6.2 A négyszög jel állapotainak táblázata:**

****

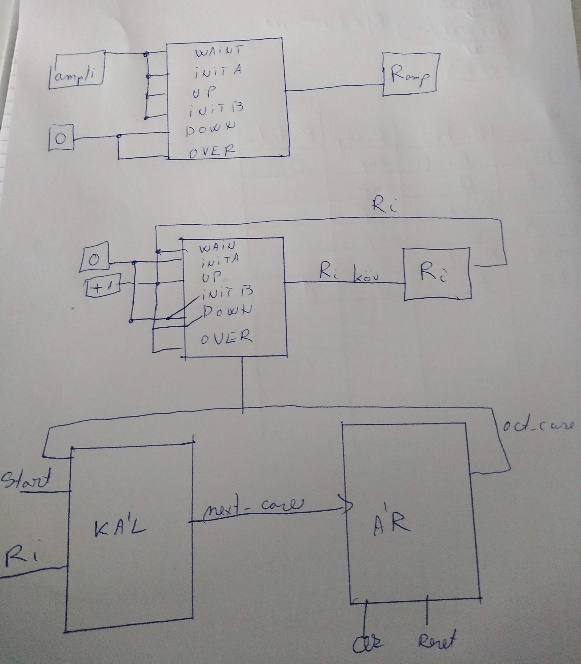
Figure

**Ri\_köv:** számláló, amint az INIT\_A-ban inicializálunk. Az UP állapotban a számlálót növelem, mig az INIT\_B-ben már le is nullázom. A DOWN állapotban megint növelem.

**R\_amp:** Ebben a helyzetben nincs szükségem következő amplitudóra, mivel ha belegondolunk, egy négyszögjelnél az amplitudó stagnál, mindig statikus értéke van. Minden állapotban egy értéke van, egész a DOWN állapotig, amikor is lenullázzuk, hogy megrajzolódjon a négyszögjel bal oldali lemenő ága. Az amplutudóból leesünk 0-ba és máris megrajzoltuk a lemenő vonalat.

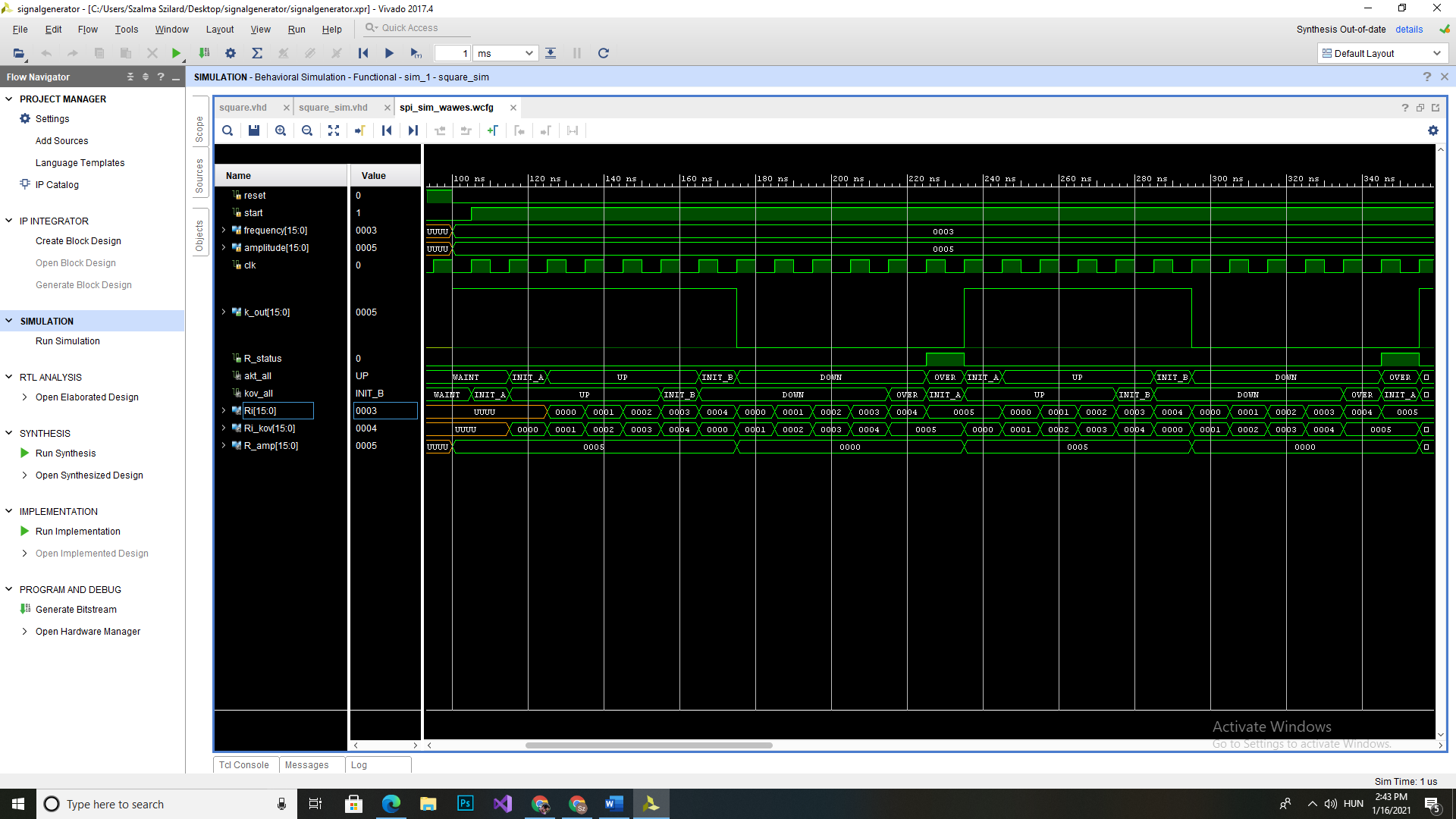
**R\_status:** Értéke statikus, egészen az OVER állapotig, amikor is 1-be vált, jelzi, hogy a négyszögjel kirajzolódott.

**6.3 A négyszög jel áramköri rajza**

****

Figure

**6.4 A négyszögjel szimulációja**



Figure

A négyszögjel szimulációján láthatjuk, hogy a jel az amplitudóból indul ki, és addig marad az amplitudóban, amig a frekvencia értékét eléri az Ri számlálóm. Amikor elérem a beállitott frekvenciát, a jel lezuhan 0-ba és megkezdődik a DOWN állapot. A DOWN állapot addig tart, amig a számláló megint eléri a frekvenciát. Ekkor újra felszökünk az amplitudóba és kirajzolódik az újjabb négyszög jel. Ebből látszik, hogy a frekvenciától függ hogy milyen hosszú a négyszögjel, és az is függ tőle, hogy a jelek milyen gyakran követik egymást.

**7. Összegzés**

A tervben sikerül létrehozni egy SPI interfészt és két különálló jelgenerátort: háromszög és négyszög jelgenerátort.

**8. További megoldandó feladatok:**

A háromszög és a négyszög jel kiküldése az SPI-n keresztül.